Axiom-based ontology matching: a method and an experiment

Frederic Furst, Francky Trichet

Axiom-based ontology matching: a method and an experiment

Frédéric Fürst, Francky Trichet

Laboratoire d'Informatique de Nantes-Atlantique
2, rue de la Houssinière
B.P. 92208
44322 NANTES CEDEX 3

— Knowledge Engineering —

RESEARCH REPORT

N° 05.02

Mars 2005

Frédéric Fürst, Francky Trichet

Axiom-based ontology matching: a method and an experiment

30 p.

Les rapports de recherche du Laboratoire d'Informatique de Nantes-Atlantique sont disponibles aux formats PostScript® et PDF® à l'URL :
http://www.sciences.univ-nantes.fr/lina/Vie/RR/rapports.html

Research reports from the Laboratoire d'Informatique de Nantes-Atlantique are available in PostScript® and PDF® formats at the URL:
http://www.sciences.univ-nantes.fr/lina/Vie/RR/rapports.html

Axiom-based ontology matching: a method and an experiment

Frédéric Fürst, Francky Trichet

Abstract

Managing multiple ontologies is now a core question in most of the applications that require semantic interoperability. The Semantic Web is surely the most significant application of this report: the current challenge is not to design, develop and deploy domain ontologies but to define semantic correspondences among multiple ontologies covering overlapping domains. In this paper, we introduce a new approach of ontology matching named *axiom-based ontology matching*. As this approach is founded on the use of axioms, it is mainly dedicated to heavyweight ontology, but it can also be applied to lightweight ontology as a complementary approach to the current techniques based on the analysis of natural language expressions, instances and/or taxonomical structures of ontologies. This new matching paradigm is defined in the context of the Conceptual Graphs model (CG), where the projection (*i.e.* the main operator for reasoning with CG which corresponds to homomorphism of graphs) is used as a means to semantically match the concepts and the relations of two ontologies through the explicit representation of the axioms in terms of conceptual graphs. We also introduce an ontology of representation, called MetaOCGL, dedicated to the reasoning of domain ontology at the meta-level.

Categories and Subject Descriptors: I.2.4 [**Knowledge Representation Formalisms and Methods**]: Representation Languages

General Terms: Knowledge Engineering, Knowledge Representation, Ontology Matching

Additional Key Words and Phrases: Heavyweight Ontologies, Conceptual Graphs

Contents

1 Introduction

Ontologies have become increasingly common on the Web where they are used to deal with the problem of data and information heterogeneity. But this first level of heterogeneity is currently accompagnied with a second level of heterogeneity related to semantics in the sense that the distributed nature of ontology development has led to a large number of ontologies covering overlapping domains. In other words, it can exist several ontologies (which have been defined by different communities) for the same domain. In this context, finding correlation between modelling primitives (*i.e.* concepts and relations) in separate ontologies becomes very important. This task, which is called ontology matching, ontology mapping or ontology alignment, is at the heart of the multiple-ontology management process that is now a core question in most of the applications that require semantic interoperability such as the Semantic Web.

The strategies for matching ontologies are quite diverse: hierarchical clustering techniques [3], Formal Concept Analysis [23]), analysis of terminological features of concepts and relations (*i.e.* names or natural-language definitions) or analysis of structure [24]. However, as recall in [11], most of the works that deals with ontology alignment only consider lightweight ontology (*i.e.* an ontology composed of a taxonomy of concepts and a taxonomy of relations). No current tool provides mapping functionalities based on other ontology components and in particular axioms which are the main building blocks for fixing the semantic interpretation of the concepts and the relations of an ontology [22]. This situation is clearly explained in the main recommendations of the OntoWeb project [17]: most of the current work related to ontology matching are only based on criteria related to the hierarchy of concepts and relations (and thus, they are not efficient for heavyweight ontologies) because most of the domain ontologies included in the libraries (available on the web) are lightweight ontologies[1].

The work presented in this paper aims at defining a new ontology matching approach based on the explicit use of all the components of a heavyweight ontology. This approach requires the explicit representation of the axioms of the two ontologies (that are considering for the matching process) at the conceptual level, and not at the operational level as it is usually the case in most of the works related to ontological engineering: for instance in Protégé [16], the axioms are directly represented in an operational form (*i.e.* a rule or a constraint with fixed and predefined operational semantics) by using the PAL language based on logical expressions. It is important to underline that since the beginning of ontological engineering, domain axioms have always being considered and represented at an operational level. As the operational form of an axiom can not easily be matched (semantically speaking) with another one, most of the current mapping techniques only use the taxonomy of concepts and/or the taxonomy of relations, and thus can not take all the semantic wealth of heavyweight ontologies into account. Our work aims at filling this gap by allowing (1) the representation of the axioms at the conceptual level[2] and (2) the use of this increase in semantics for concepts and relations mapping.

To represent heavyweight ontologies at the conceptual level, we use OCGL (*Ontology Conceptual Graphs Language*) [9]. This modelling language is based on a graphical syntax inspired from those of the Conceptual Graphs model (CGs)[3]. It allows us to represent terminological knowledge through the specification of concepts and relations, and to represent both classical properties (such as subsomption or algebraic properties) and any kind of axioms at the conceptual level. This explicit graph-based representation of axioms coupled with reasoning capabilities based on graphs homomorphism facilitates the topological comparison of axioms. The method we propose mainly relies on this feature: ontology morphism founded on graph-based knowledge representation and graph-based reasoning mechanisms.

The rest of this paper is organized as follows. Section 2 presents the modelling paradigm we advocate for

[1] The more significant example of this situation is the benchmark used during the Ontology Alignment Contest at the 3rd Evaluation of Ontology-based Tools (EON'2004) Workshop (`http://km.aifb.uni-karlsruhe.de/ws/eon2004`).

[2] At the conceptual level, an axiom has a formal semantics but not an operational one. At the operational level, an axiom has both a formal and an operational semantics, and this latter clearly limits its reuse and thus the reuse of the ontology. The operational semantics of an axiom, represented through a set of rules and/or constraints, expresses the way the axiom is used to reason, whereas the formal semantics expresses the way the axiom constrains the interpretation of the primitives (*i.e.* the concepts and the relations which are involved in the axiom).

[3] The Conceptual Graphs model, first introduced by Sowa [20], is an operational knowledge representation model which belongs to the field of semantic networks. This model is mathematically founded both on logics and graph theory [20]. Two approaches for reasoning with CGs can be distinguished: (1) considered CGs as a graphical interface for logics and reasoning with logic and (2) considered CGs as a graph-based knowledge representation and reasoning formalism with its own reasoning capabilities. In our work, we adopt the second approach by using the projection (a graph-theoretic operation corresponding to homomorphism) as the main reasoning operator; projection is sound and complete w.r.t. deduction in FOL.

defining a domain ontology. Section 3 presents the domain we consider in this paper for evaluating our work. Section 4 introduces the basic fundations of our axiom-based matching method and presents the principles of our algorithm. Section 5 comments the results of an experimentation in the context of a simple domain related to family relationships. Section 7 compares our approch with related work and introduces our current study which aims at defining an approach for reasoning domain ontolgies at the meta-level, for instance for verification/validation purposes.

2 Context of the work: the OCGL modelling language

The OCGL modelling language (*Ontology Conceptual Graphs Language* [9]) we advocate for specifying an ontology (at the conceptual level) is based on three building blocks: Concepts, Relations and Axioms. Representing an ontology in OCGL mainly consists in (1) specifying the conceptual vocabulary of the domain and (2) specifying the semantics of this conceptual vocabulary through axioms.

The conceptual vocabulary consists of a set of **Concepts**, a set of **Relations** and a set of ontological instances of concepts[4]. The sets of concepts and relations can be structured by using both well-known conceptual properties called Axiom Schemata and Domain Axioms.

The **Axiom Schemata** proposed in OCGL are:

1. the *ISA* link between two concepts or two relations (subsomption property) used to construct concept/relation taxonomies (tree or lattice);
2. the *Abstraction* of a concept (which corresponds to an *Exhaustive-Decomposition* in some works [11]);
3. the *Disjunction* between two concepts[5];
4. the *Signature* of a relation;
5. the *Algebraic properties* of a relation (symmetry, reflexivity, transitivity, irreflexivity, antisymmetry);
6. the *Exclusivity* or the *Incompatibility* between two relations[6];
7. the *Cardinalities* (Maximale and Minimale) of a relation.

OCGL has been implemented in a tool, called TooCoM (*a Tool to Operationalize an Ontology with the Conceptual Graph Model*), dedicated to the edition and operationalization of domain ontologies [8][7]. Thanks to this tool, it is possible to define the conceptual primitives (concepts and relations) and to specify the axiom schemata in a graphical way. Figure 2 (resp. figure 3) shows the hierarchies of concepts (resp. relations) of two ontologies dedicated to family relationships.

Domain Axioms differ from axiom schemata in the sense that they are totally specific to the domain whereas axiom schemata represent classical properties of concepts or relations. The OCGL graphical syntax used to express such an axiom is based on the Conceptual Graphs model. Thus, an axiom is composed of an *Antecedent part* and a *Consequent part*, with a formal semantics that intuitively corresponds to: *if the Antecedent part is true, then the Consequent part is true*. Figure 1 shows the OCGL graph representing the axiom "*The enemy of my friend is my enemy*".

Remark: Since many existing knowledge-representation systems (*e.g.* Frame-based or DL-based systems) are compatible with the OCGL modelling language, the solution we propose can be applied to a variety of knowledge-representation systems.

[4] An ontological instance of a concept is an instance required to express the semantics of the domain. For example, in the domain of mathematics, π is an ontological instance of the concept *Number*, because the expression of many axioms of this domain requires this instance. But 3.54 is not an ontological instance.

[5] Note that it is possible to define a *Partition* [11] by using the abstraction and the disjunction. For instance, the decomposition of `Number` into (`OddNumber` and `EvenNumber`) is a partition because `Number` is an abstract concept and `OddNumber` and `EvenNumber` are disjoint.

[6] The incompatibility between two relations R_1 and R_2 is formalized by $\neg(R_1 \wedge R_2)$, the exclusivity is formalized by $\neg R_1 \Rightarrow R_2$.

[7] TooCoM is available under GNU GPL license at `http://sourceforge.net/projects/toocom/`.

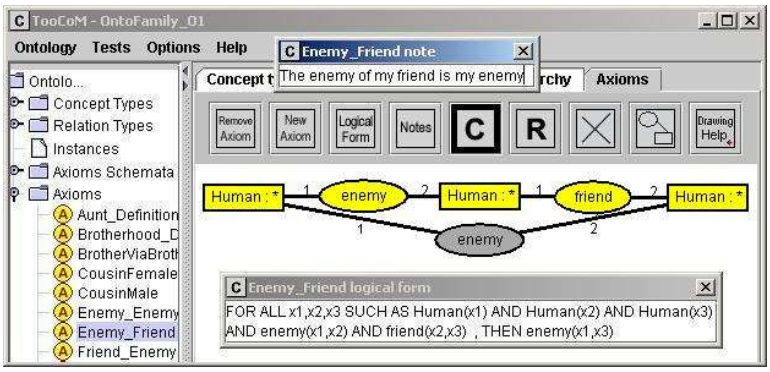

Figure 1: Representation of an axiom in TooCoM. The bright nodes represent the antecedent part, the dark ones the consequent part. Each part contains concept nodes (indicated by rectangles) and relation nodes (indicated by ellipses). A concept node is described by a label and a marker that identifies the considered instance (the marker $*$ denotes an undefined instance). A relation node is only described by a label. An edge between a concept and a relation is labeled with the position of the concept in the signature of the relation. The logical expression of the graph is automatically generated.

3 Domain of the experiment: OntoFamily

In order to illustrate our ideas, this paper considers a very simple (but intuitive) domain related to family relationships. This limited domain includes the following notions[8]: *father, mother, grandfather, grandmother, son, daughter, cousin, nephew, niece, uncle, aunt, sister, brother, wife, husband, friend, enemy*. This example is interesting in so far as it is easy to understand and also because it necessarily requires Domain Axioms for defining some notions (for instance, "*An aunt is either a female sibling of one of one's parents or the wife of an uncle who is the male sibling of a parent*") or specifying relations between notions (for instance, "*The enemy of my enemy is my friend*"). In other words, Axioms Schemata are not sufficient for representing all the knowledge of this domain.

In the context of a course on Ontological Engineering, two groups of students in Master degree have worked separately to construct (by hand) an ontology of this limited domain. This experiment has led to the definition (and the representation in OCGL) of two ontologies respectively called OntoFamily O_1 and OntoFamily O_2[9].

3.1 Overview of OntoFamily O_1

OntoFamily O_1 is composed of:

- 3 types of concepts (cf. figure 2) which define a partition: `Human` (an abstract concept) and its two subconcepts `Man` and `Woman` which are disjoint;
- 31 binary relations structured into a lattice (lattice-depth=3) (*cf.* figure 3);
- 11 axioms shemata (cf. figure 4): 1 for the abstraction, 1 for disjunction, 1 for exclusivity, 3 for the symmetry, 4 for the cardinalities and 1 for the transitivity. Note that the abstractions and the disjunctions or exclusivities are directly managed by the TooCom interface, this is why the list (in the figure) only includes 8 axioms schemata;
- 18 axioms (cf. figure 5 and figure 6 for examples).

[8]We voluntary use the term *notion* in order to avoid confusion with concept or relation. Indeed, we only present here what must be considered in the domain and not how it must be considered (i.e. the modelling choice between a concept or a relation).

[9]These ontologies are available in CGXML at `http://sourceforge.net/projects/toocom/`. CGXML is the XML storage format used for OCGL.

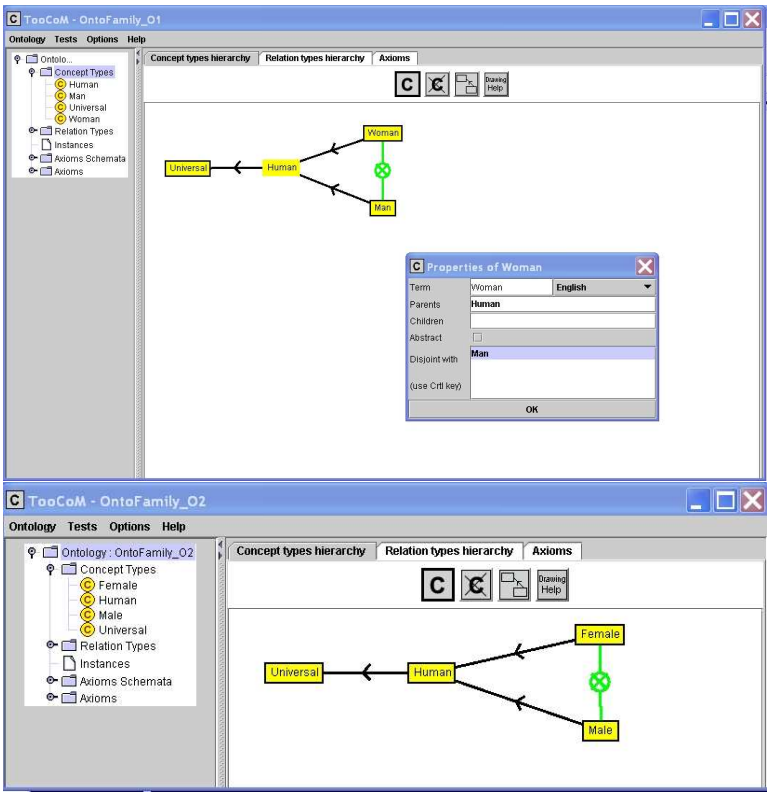

Figure 2: Concepts of OntoFamily O_1 and OntoFamily O_2. An arrow represents a subsomption link between a concept and one of its parent. A concept without surround is abstract. The crossed circles represent disjunctions between concepts.

3.2 Overview of OntoFamily O_2

OntoFamily O_2 is composed of:

- 3 types of concepts (cf. figure 2) : `Human` (which is not abstract) and its two sub-concepts `Male` and `Female` which are disjoint;
- 23 binary relations (cf. figure 3) structured into a tree (tree-depth=2);
- 10 axioms schemata (cf. figure 4): (1 for disjunction, 1 for exclusivity, 3 for the symmetry, 4 for the cardinalities and 1 for the transitivity);
- 27 axioms (cf. figure 5 for an example).

3.3 Comparaison of OntoFamily O_1 and OntoFamily O_2

Table 1 summarizes the differences (from a quantitative point of view) between OntoFamily O_1 and OntoFamily O_2.

4 Axiom-based semantic matching

The objective of ontology matching is to discover and evaluate identity links between conceptual primitives (concepts and relations) of two given ontologies supposed to be built on connected domains. Our approach relies on the

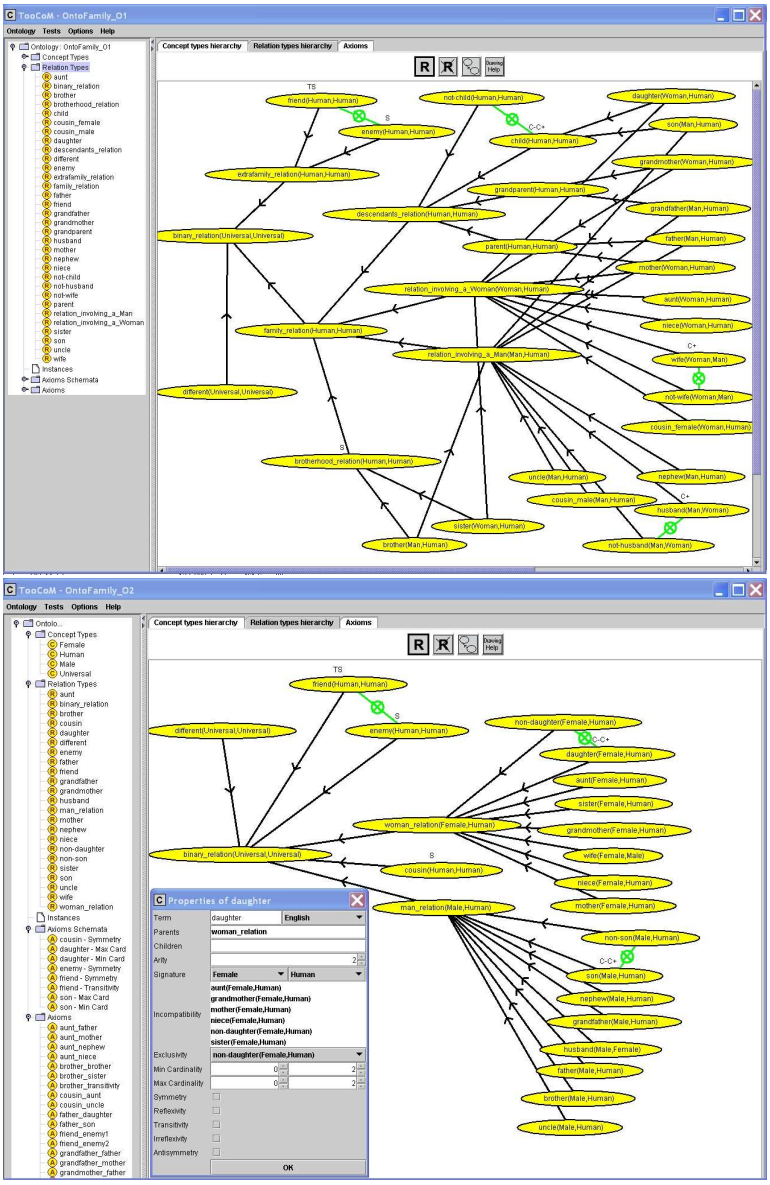

Figure 3: Relations of OntoFamily O_1 and OntoFamily O_2. An arrow represents a subsomption link. A crossed circle represents an incompatibility (or exclusivity) between two relations. Algebraic properties and cardinalities of a relation are indicated by symbols above the name of the relation (S for symmetry, T for transitivity, C+ and C- for the cardinalities, etc.).

use of the axiomatic level of the ontologies to discover semantic analogies between primitives, in order to reveal identities between them and to calculate the similarity coefficient of these identities, *i.e.* a coefficient that indicates how closely two concepts or relations are related. Of course, using the axiomatic level does not forbid to use the terminological level; these two approaches complement each other.

Two principles govern our method: (1) the use of the *modelling stability* and the *rarity* of a conceptual property to fix its weight for the evaluation of matchings and (2) the use of *meta-representations* of domain axioms to compare their structures.

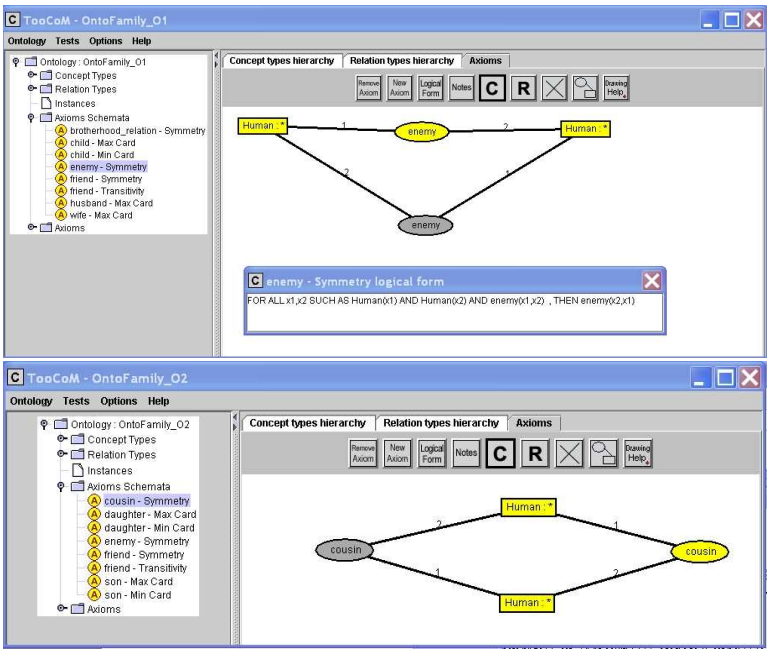

Figure 4: Axioms Schemata of OntoFamily O_1 and OntoFamily O_2 (automatically generated by TooCom).

	Concepts		Relations (binary)		Axiom Schemata	Domain Axioms
	Nb	Structure	Nb	Structure		
O_1	3	tree depth = 2	31	lattice depth = 3	*1 abst* *1 disj* *1 exclu* *3 sym* *4 card* *1 trans*	18
O_2	3	tree depth = 2	23	tree depth = 2	*1 disj* *1 exclu* *3 sym* *4 card* *1 trans*	27

Table 1: Quantitative differences between O_1 and O_2. Note that signatures of relations are not explicitly indicated but they are of course specified in both ontologies.

4.1 Assumption about stability and rarity

As recalled in [6], when dealing with heterogeneous knowledge resources, one key issue is understanding what forms of heterogeneity exist between the knowledge sources and what are the mismatches they can cause. We can broadly distinguish between mismatches caused by *non-semantic* and *semantic heterogeneity*. The former type of heterogeneity is also known as syntactic or language heterogeneity, while the latter is also called ontology heterogeneity. Syntactic heterogeneity denotes the differences in the language primitives that are used to specify ontologies, while semantic heterogeneity denotes differences in the way the domain is conceptualised and modelled. In our work, we only consider *semantic heterogeneity*, since the two ontologies that are considered for the matching are supposed to be represented in the same knowledge representation language: OCGL. Mismatches caused by *semantic heterogeneity* occur when different ontological assumptions are made about the same domain. In [6], six types of mismatches are introduced (*Representation paradigm*, *Top-level concepts*, *Modelling convention*, *Synonym terms*, *Homonym terms* and *Encoding*). The most well-known are the modelling convention mismatches which depend on modelling decisions made while designing the ontology. For instance, it is often the case that an ontology designer has to decide whether to model a certain distinction by introducing a new concept or a new relation. These mismatches clearly demonstrate that the stability of the *ISA* taxonomies of concepts (resp. relations) is not really strong. In order to generalize this phenomena, we make assumptions about the stability and

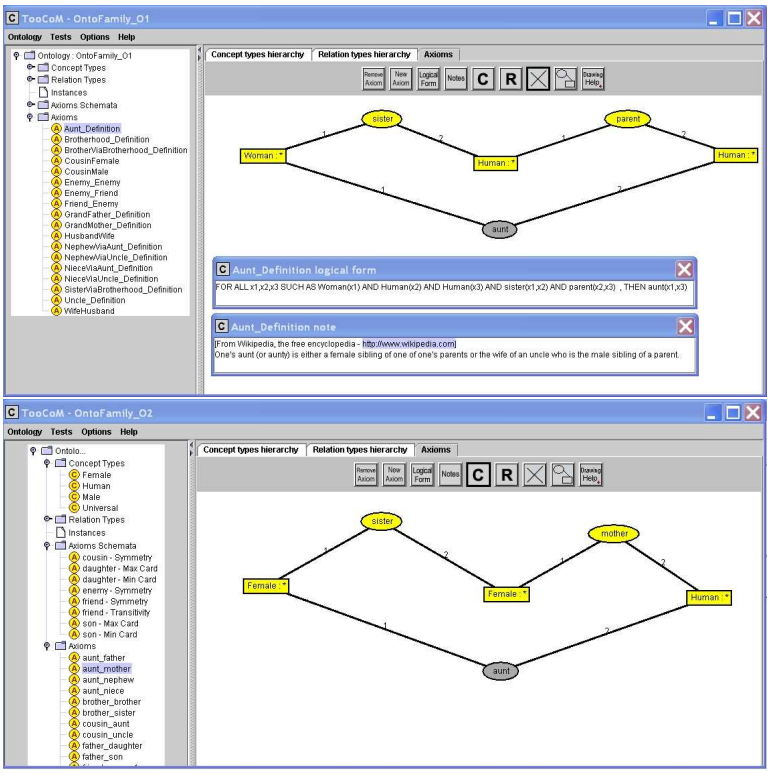

Figure 5: Axiom "Aunt" of OntoFamily O_1 and axiom "AuntMother" of OntoFamily O_2. O_2 needs more than one axiom for representing this notion because of the modelling choice, in particular the fact that the notions *Child* and *Parent* are not considered in O_2. Note that the logical expression of an axiom is automatically generated by TooCom. The windows called "Note" enable the storage of natural language definitions.

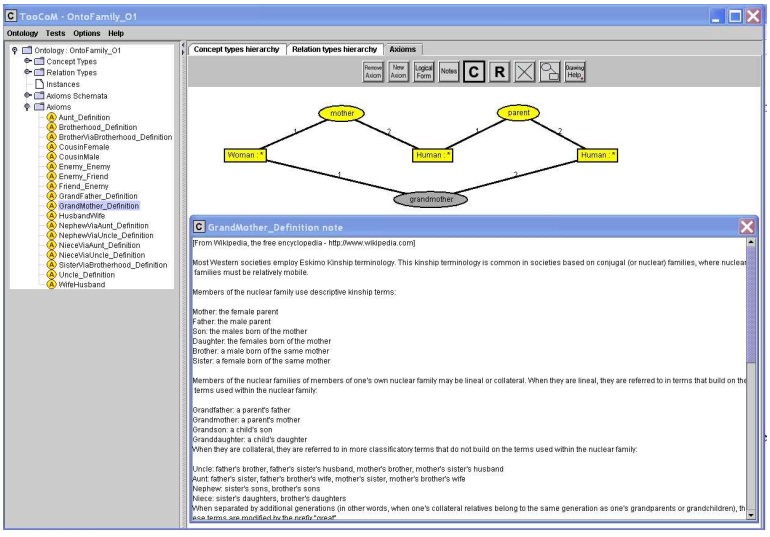

Figure 6: Axiom "GrandMother" of OntoFamily O_1.

the rarity of the OCGL properties used to model a domain.

Conceptual properties of a domain, expressed in OCGL by axiom schemata and domain axioms, can be mod-

elized in different ways. For instance, two hierarchies of concepts can be different if additional concepts are added to structure them or not (*e.g.* the *Human* concept could have been omitted in OntoFamily). In a similar way, the signature of a relation can be different from a modelization to another one (*e.g.* the notion of *cousin* can be represented by only one relation *cousin(Human,Human)*, or by two relations when introducing the gender *cousin(Woman,Human)* and *cousin(Man,Human)*). The modelling stability of a property (*i.e.* an axiom schema or a domain axiom) indicates its degree of stability from an ontology to another one. The search of correspondences between ontologies must favor the properties which own the higher modelling stability, because analogies between them in the two ontologies are more relevant than others.

Moreover, in the same ontology, a property can be very common or, on the contrary, extremely rare. For example, the symmetry property is very common. But a domain axiom is, by definition, very particular and only a few axioms with the same formal semantics can exist in a given ontology. The rarity of a property in the ontologies to align increases the weight of the matchings discovered thanks to this property. So, at the beginning of the matching process, these rarities must be valuated in order to adapt the weight of each property to the characteristics of the ontologies to align.

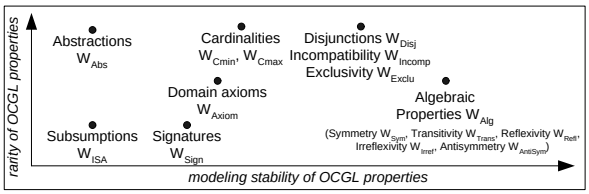

Figure 7: modelling stability and rarity of OCGL properties. Subsomptions and signatures are both very common and not very stable. Domain axioms and algebraic properties are less common, and the stability of domain axioms is lower than those of algebraic properties because the representations of algebraic properties are fixed. Cardinalities are less stable than properties that link two concepts or relations (*i.e.* disjunction, incompatibility and exclusivity), because both their values and the relations they are about can change.

Figure 7 presents the relative values of stability and rarity of the OGCL properties. These evaluations of modelling stability and rarity are only assumptions that fix the default weights of the modelling properties. These weights can be modified by the user, in order to improve the results of each matching process. By default, the values of the weights are ordered as follows: $W_{Alg}(W_{Sym}, W_{Trans}, W_{Refl}, W_{Irref}, W_{AntiSym}) > W_{Disj} = W_{Incomp} = W_{Exclu} > W_{Cmin} = W_{Cmax} > W_{Axiom} > W_{Sign} > W_{Abst} > W_{ISA}$.

The less valuated properties are the subsomption links between primitives, because they are very common in ontologies, and different hierarchies can easily be built for the same domain (for example by adding abstract concept to structure them). Signatures of relations are also very common but their variability from an ontology to another is lower. Domain axioms and algebraic properties of relations are less common than subsumption links and signatures. The stability of domain axioms is lower than those of algebraic properties, because the representation of algebraic properties are fixed. The axioms are built by the user and several axioms can be built to modelize the same properties, for example by specifying, or not, *difference* relation between concepts. The other properties (abstraction, cardinality, disjunction, incompatibility/exclusivity) are even rarer than the previous one. But abstraction seems to be a more variable property than cardinalities, and cardinalities seem to be less stable than properties that link two concepts or relations (disjunction, incompatibility/exclusivity), because both their values and the relations they are about can change.

4.2 MetaOCGL: an ontology of representation

To detect analogies between axioms represented as graphs, and then to detect analogies between the primitives corresponding to the nodes of the graphs, the domain axioms are transcribed into a more abstract form, that preserves the topological structures of the graphs. These abstract representations are based on an ontology of the OCGL language, expressed in OCGL, and called **MetaOCGL**. As shown in figure 8, MetaOCGL includes all the concepts of OCGL and their relations (*isa* relation, exclusivity/incompatibility between relations, disjunction of

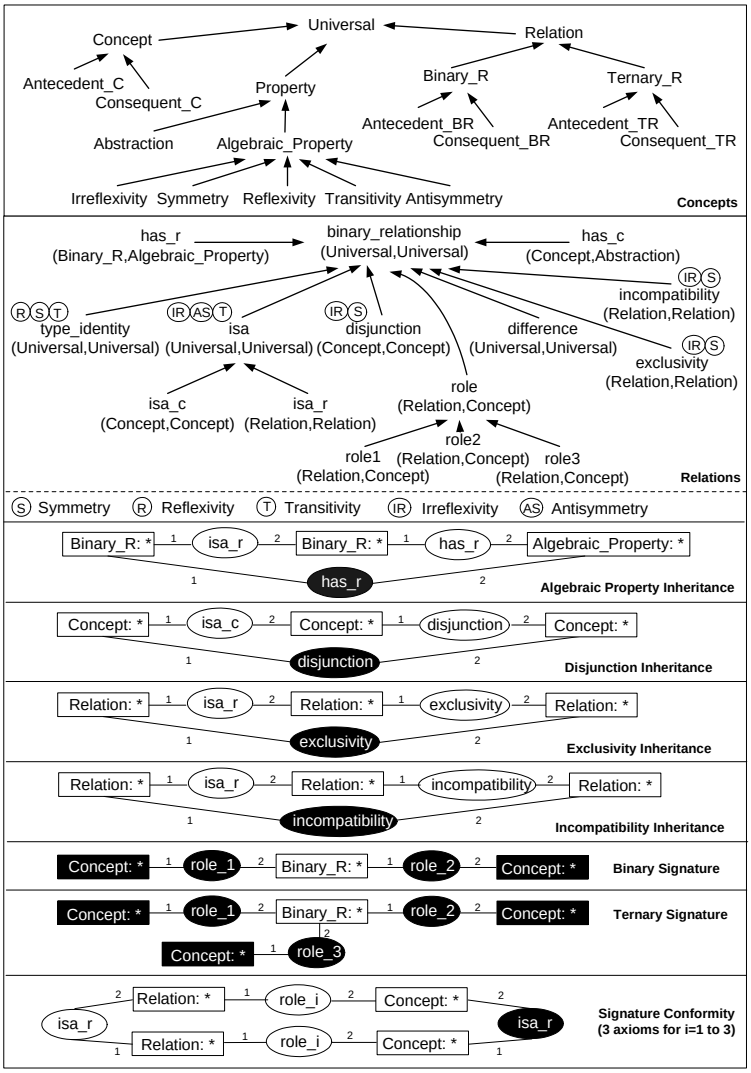

Figure 8: Concepts, relations, axiom schemata and domain axioms of MetaOCGL.

concepts, links between relations and concepts in a graph that expresses an axiom). Figure 9 shows the OCGL graphs dedicated to the representation of the two axioms of OntoFamily O_1 "*the enemy of my enemy is my friend*" and "*the enemy of my friend is my friend*", and their corresponding meta-graphs in MetaOCGL.

The comparisons between axioms represented in MetaOCGL are performed by using the *projection* operator of the Conceptual Graphs model, a graph-theoretic operation corresponding to homomorphism which is sound and complete w.r.t. deduction in FOL. Given two graphs G_1 and G_2, which represent in MetaOCGL two axioms A_1 and A_2, if two projections exist from G_1 into G_2 and from G_2 into G_1, then A_1 and A_2 have the same structure. In this case, the axioms A_1 and A_2 express the same type of property, and the analogy between the two axioms can be extended to the primitives that appear in the axioms.

4.3 Algorithm

Our algorithm takes as input two ontologies O_1 and O_2 (represented in OCGL) and provides as output potential similarity between two concepts or two relations: the result is a set of matchings (P_i, P'_j, C), where P_i and P'_j are respectively conceptual primitives (concepts and relations) of O_1 and O_2, and C the similarity coefficient between

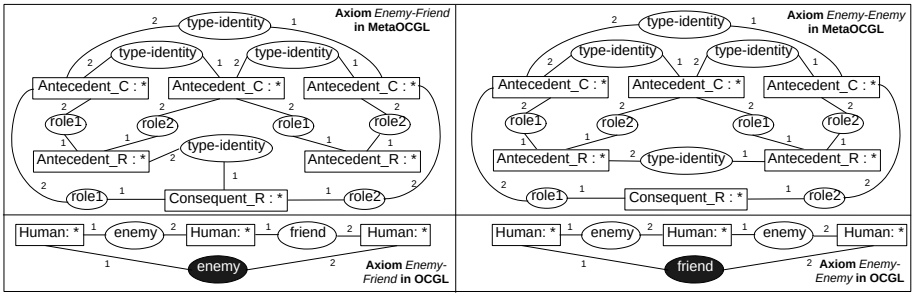

Figure 9: Two axioms of OntoFamily represented with MetaOCGL. The *type_identity* links denote that the nodes are of the same type in the axiom.

P_i and P'_j. Of course, for a given primitive P_i of O_1, several (or any) matchings can exist with primitives of O_2, and vice versa. Both axiom schemata and domain axioms are used to evaluate or discover primitive matchings. Of course, the weight of each OCGL property is used to modulate its influence on the evaluation of the matching. Again, these weights are parameters of our algorithm which can be changed to improve the precision of the results.

4.3.1 Using axiom schemata to evaluate matchings

Axiom schemata that deal with only one primitive (*i.e.* algebraic properties and abstractions) are compared from O_1 to O_2, in order to discover primitive matchings. If an algebraic property (resp. an abstraction) appears in O_1 for a primitive p_1 and in O_2 for a primitive p_2, the coefficient c of the matching (p_1, p_2, c), if it exists, is increased by W_{Alg} (resp. W_{Abs}). If the matching does not exist, (p_1, p_2, W_{Alg}) (resp. (p_1, p_2, W_{Abs})) is created. If an algebraic property (resp. an abstraction) appears in O_1 for p_1 but not in O_2 for p_2 (or inversely), the coefficient c of the matching (p_1, p_2, c), if it exists, is decreased by W_{Alg} (resp. W_{Abs}). If it does not exist, $(p_1, p_2, -1 * W_{Alg})$ (resp. $(p_1, p_2, -1 * W_{Abs})$) is created. A partition[10] is a property which is more semantically rich than a simple abstraction. So, if two concepts c_1 and c_2 are respectively the head concept of a partition in O_1 and O_2, the coefficient c of the matching (c_1, c_2, c), if it exists, is increased by $2 * W_{Abs}$ (or decreased by $2 * W_{Abs}$ if only one concept is involved in a partition). If it does not exist, $(c_1, c_2, 2 * W_{Abs})$ (or $(c_1, c_2, -2 * W_{Abs})$) is created.

Axiom schemata that deal with two primitives (*i.e.* disjunction, incompatibility and exclusivity) are used either to modify the coefficients of existing matchings, or to create new ones. The coefficient of a matching whose two primitives are involved in a disjunction, an incompatibility or an exclusivity is increased by the corresponding weight (*i.e.* W_{Disj}, W_{Incomp} or W_{Exclu}). It is decreased if only one of the primitive is part of such a property. The matching is created with the corresponding coefficient if it does not exist.

Finally, table 2 presents the different actions that are done when considering the cardinalities. If the matching between the two considered relations does not exist when an analogy between cardinalities is found, the matching is created, with the corresponding coefficient. Only cardinalities of relations with the same arity are compared.

4.3.2 Using domain axioms to evaluate matchings

As introduced in section 4.2, domain axioms are represented in MetaOCGL in order to compare their structures. For each axiom couple (a_1, a_2), where $a_1 \in O_1$ and $a_2 \in O_2$, the representations of a_1 and a_2 in MetaOCGL, $meta(a_1)$ and $meta(a_2)$, are built. These representations can be enriched by adding information about the nodes: for instance, in figure 9, the two relations *enemy* of the axiom in OCGL are represented in MetaOCGL by the two concepts *Antecedent_R* which are linked by the meta-relation called *type_identity*.

Two types of topological equivalence are then considered:

[10] A *partition* [11] is the combination of the abstraction of a concept (the head) and the disjunction between its children. For instance, the decomposition of *Number* into (*OddNumber* and *EvenNumber*) is a partition because *Number* is an abstract concept and *OddNumber* and *EvenNumber* are disjoint.

	Relation r_1 in Ontology 1	Relation r_2 in Ontology 2	Action
Min Card	0 (resp. $c_{min} \geqslant 1$)	$c_{min} \geqslant 1$ (resp. 0)	$-2 * W_{cmin}$
	$c_{min} \neq 0$	$c_{min} \neq 0$	$+2 * W_{cmin}$
	$c1_{min} \neq 0$	$c2_{min} \neq 0$ and $\neq c1_{min}$	$-W_{cmin}$
	Relation r_1 in Ontology 1	Relation r_2 in Ontology 2	Action
Max Card	∞ (resp. $c_{max} \geqslant 1$)	$c_{max} \geqslant 1$ (resp. ∞)	$-W_{cmax}$
	$c_{max} \neq \infty$	$c_{max} \neq \infty$	$+2 * W_{cmax}$
	$c1_{max} \neq \infty$	$c2_{max} \neq \infty$ and $\neq c1_{max}$	$-2 * W_{cmax}$

Table 2: Modifications of the coefficient of the matching (r_1, r_2, c) according to the cardinalities of the relations. c_{min} and c_{max} are the values of cardinalities for the relations (for a given element of their signatures).

1. the **Equivalence**, that occurs when projections exist from $meta(a_1)$ to $meta(a_2)$ and from $meta(a_2)$ to $meta(a_1)$, without considering the $type\_identity$ relations;
2. the **Typed Equivalence** that occurs when the two projections exist with the $type\_identity$ relations.

The weight of a typed equivalence is higher than those of an equivalence. A typed equivalence (resp. equivalence) between two axioms increases the coefficient of nodes linked by projection by the weight of the axiom typed equivalence (resp. equivalence). When no projection (or only one) exists, no modification is done. For example, the two domain axioms of figure 9 are equivalent because two projections exist between their meta-graphs without considering the *type-identity* relations. When considering the *type-identity* relations, there exists no projection, so they are not typed equivalent.

4.3.3 Resolving matchings

Because they link a relation and a set of concepts, signatures of relations are only used to increase or decrease existing matching coefficients, and not to create new ones. However, two contexts of use are distinguished: modifying relation matching coefficients by considering the matchings between concepts of their signatures, or modifying concept matching coefficients by considering the matchings between relations with the considered concepts in their signatures. For example, if the relations *mother1(Woman1,Human1)* and *parent2(Human2,Human2)* match, the coefficients of the matchings *(Woman1,Human2)* and *(Human1,Human2)*, if they exist, are increased by W_{Sign}. But if the matchings *(Woman1,Human2)* and *(Human1,Human2)* exist, the coefficient of the relation matching *(mother1,parent2)* can also be increased by W_{Sign}. The choice between these two possibilities is based on the ratio $Found_C$ (or $Found_R$) between the number of concept (or relation) matchings and the average number of concepts (or relations) in O_1 and O_2: if $Found_C$ is higher than $Found_R$ (that is the proportion of discovered concept matchings is higher than those of relation matchings), the signatures are used to improve the coefficients of existing relation matchings with the existing concept matchings, and vice-versa. So, before using the signatures, the concept (or relation) matchings must be fixed, then the signatures are used to improve the relation (or concept) matchings, which are then fixed.

Fixing concept matchings or relation matchings is done by considering their coefficients: when several matchings exist for one primitive to link, the matching with the higher coefficient is considered as the most probable. Of course, the coefficients do not totally prevent contradictions between 3 or more matchings. In this case, the user must choose one of the proposed matchings.

5 Experimental results

5.1 Assessment

The application of our algorithm on OntoFamily O_1 and OntoFamily O_2 (before the matching resolving step) leads to 201 matchings, including 9 concept matchings (among 9 possible ones), and 192 relation matchings (among 713 possible ones). A lot of these matchings only appear in three or less axiom comparisons, and never in axiom

Primitive of ontology O_1	Primitive of ontology O_2	Axioms		Multiple Primitives Properties			*ISA* relations	Abstraction
		W_{Topo}	W_{Topo+}	W_{Disj}	W_{Incomp}	W_{Exclu}	W_{isa}	W_{Abst}
Human	**Human**	98	10	0	/	/	0	-1
Human	Male	84	10	0	/	/	0	-1
Human	Female	76	0	0	/	/	0	-1
Woman	Human	9	3	0	/	/	0	0
Woman	Male	52	6	0	/	/	0	0
Woman	**Female**	54	6	0	/	/	0	0
Man	Human	9	3	0	/	/	0	0
Man	**Male**	52	6	0	/	/	0	0
Man	Female	56	6	0	/	/	0	0

Figure 10: Results for the concepts

schema comparison, so they are rejected because of their low level of relevance comparing to the other ones. Thus, 9 concept matchings and 69 relation matchings are considered for the resolving step (cf. figure 11 and figure 10).

In this context, $MR_c = 3$ and $MR_r \approx 2.55$, so, the concept matchings are resolved before the relation ones. The *(Human,Human)* matching has the higher score with $108 * W_{Axiom}$ (98 equivalences and 10 typed equivalences), plus $-1 * W_{Abs}$ (because *Human* is abstract in O_1 but not in O_2). For the other concepts, the matching with the higher coefficient is *(Woman,Female)*, with $60 * W_{Axiom}$ (54 equivalences and 6 typed equivalences); the coefficient of *(Man,Male)* is equal to $58 * W_{Axiom}$ (52 equivalences and 6 typed equivalences). All the links between concepts are discovered by our algorithm.

When the concept links are fixed, the signatures are used to improved the coefficient of relation matchings: the coefficient of each relation matching is possibly increased or decreased by W_{Sign} according to whether the concepts of the two relations are linked or not. Finally, the resolving matching process produces 9 relation links (among 17 true links): *(aunt,aunt)*, *(daughter, daughter)*, *(enemy,enemy)*, *(father,father)*, *(friend,friend)*, *(husband,husband)*, *(son,son)*, *(uncle,uncle)* and *(wife,wife)*.

These results, although perfectible, are encouraging according to the context of the experiment, *i.e.* two "small" heavyweight ontologies. Indeed, our approach could be more relevant in a context of more large and complex ontologies (in terms of numbers of concepts and relations and of numbers of heterogeneous axioms). Note that this experiment has also shown that using subsomption links can improve the efficiency of our algorithm. For example, the relation matchings *(parent,mother)* and *(parent,father)* have high coefficients, so it is relevant to deduce the *(parent,parent)* matching.

5.2 Detailed results

Figures 10 and 11 present respectively the concept matchings and the relation matchings which are discovered and evaluated by our algorithm. For each discovered couple of primitives, and for each property, the given value corresponds to the number of times that the property has permitted to increase or decrease the matching validity. For instance, in figure 10, the couple *(Human,Human)* has been founded 98 times in domain axiom equivalences, 10 in domain axiom typed equivalences, never in disjunction comparisons, never in subsumption comparisons, and one time in abstraction comparison (with a différence between abstraction properties, which implies a decreasing of the value). The / symbol indicates that the property is not relevant for the primitives of the couple, for instance the algebraic, or incompatibility/exclusivity properties are not relevant for concepts.

6 Generalization of our approach: reasoning ontologies at the meta-level

In this section, we claim that for reasoning on a domain ontology, it is relevant to represent it at the meta-level, in order to consider it as a knowledge base and thus to make any kind of reasoning possible, such as ontology mapping/matching, ontology merging, ontology verification, ontology validation, etc. Our approach consists in using an ontology of representation (MetaOCGL) for representing a domain ontology expressed in OCGL [9]. MetaOCGL is the ontology of representation which describes all the modeling primitives (and their relations) of OCGL and its formal semantics. This ontology is represented in OCGL and this is why we call it MetaOCGL.

Primitive of ontology O_1	Primitive of ontology O_2	Algebraic Properties					Cardinalities		Axioms		Multiple Primitives Properties			Signatures	*ISA* relations	Abstraction
		W_S	W_T	W_R	W_I	W_A	W_{cmin}	W_{cmax}	W_{Topo}	W_{Topo+}	W_{Disj}	W_{Incomp}	W_{Exclu}	W_{sign}	W_{isa}	W_{Abst}
aunt	uncle								5	3	/			-1		/
aunt	aunt								5	1	/			1		/
aunt	brother								4	1	/			-1		/
aunt	sister								4	1	/			1		/
aunt	mother								3	1	/			1		/
brother	friend	1									/		-1	-1		/
brother	enemy	1									/		-1	-1		/
brother	cousin	1									/			-1		/
brotherhood	friend	1									/		-1	1		/
brotherhood	enemy	1									/		-1	1		/
brotherhood	cousin	1									/			1		/
child	husband						-2	-2			/			-1		/
child	wife						-2	-2			/			-1		/
child	daughter							2			/			-1		/
child	son							2			/			-1		/
child	mother								8		/			-1		/
child	father								8		/			-1		/
daughter	daughter						2	2			/			1		/
daughter	son						2	2			/			-1		/
daughter	wife						-2	-2			/			-1		/
daughter	husband						-2	-2			/			-1		/
enemy	friend	1							2	2	/		1	1		/
enemy	enemy	1							3	3	/		1	1		/
enemy	mother								15		/		-1	-1		/
enemy	father								14		/		-1	-1		/
enemy	brother								10		/		-1	-1		/
enemy	sister								10		/		-1	-1		/
enemy	uncle								6		/		-1	-1		/
enemy	aunt								6		/		-1	-1		/
father	father								2	1	/			1		/
friend	friend	1	1						1	1	/		1	1		/
friend	enemy	1							2	2	/		1	1		/
friend	cousin	1							2		/		-1	1		/
friend	mother								6		/			-1		/
friend	sister								5		/			-1		/
friend	brother								5		/			-1		/
husband	wife							2	2		/			-1		/
husband	husband							2	2		/			1		/
husband	son						-2		2		/			-1		/
husband	daughter						-2		2		/			-1		/
mother	father								2	1	/			-1		/
nephew	nephew								3	1	/			1		/
nephew	son								3	1	/			1		/
nephew	daughter								3	1	/			-1		/
nephew	brother								4	1	/			1		/
niece	nephew								3	2	/			-1		/
niece	sister								6	1	/			1		/
niece	brother								4	1	/			-1		/
niece	niece								3	1	/			1		/
niece	son								3	1	/			-1		/
parent	mother								19		/			-1		/
parent	father								16		/			-1		/
sister	friend	1									/		-1	-1		/
sister	enemy	1									/		-1	-1		/
sister	cousin	1									/			-1		/
son	daughter						2	2			/			-1		/
son	son						2	2			/			1		/
son	wife						-2	-2			/			-1		/
son	husband						-2	-2			/			-1		/
uncle	aunt								5	1	/			-1		/
uncle	uncle								5	1	/			1		/
uncle	brother								4	1	/			1		/
uncle	father								3	1	/			1		/
uncle	father								3	1	/			-1		/
wife	wife							2	2		/			1		/
wife	husband							2	2		/			-1		/
wife	son						-2		2		/			-1		/
wife	daughter						-2		2		/			-1		/

Figure 11: Results for the relations

6.1 Operationalization: basic fundations

Ontologies can be used to shared knowledge between systems or between systems and humans, to reason on knowledge bases or to search in knowledge bases. Thus, they have to integrate all the knowledge of a given domain, and not only the terminological knowledge, as in a thesaurus. Ontologies have to evolve from *lightweight ontologies*, which only include some well-known properties such as subsomptions or algebraic properties, to *heavyweight ontologies*, which include all axioms that are needed to represent the semantics of the domain [22].

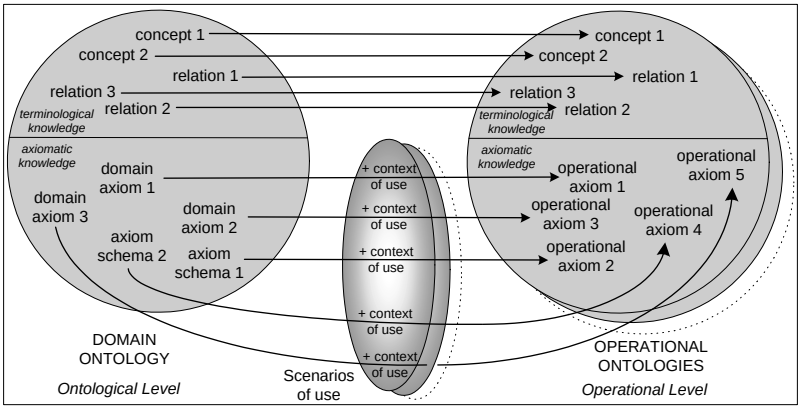

Figure 12: The operationalization process of an heavyweight ontology. Terminological knowledge is represented in the same way at the ontological level and the operational level. The representation of axiomatic knowledge at the operational level depends on a scenario of use that describes the way the axioms are used to reason in the operational ontology. The operational representation of an axiom is a set of rules and/or constraints.

But, for keeping the independence of an ontology from the applications where it is used, in order to ensure its portability, the representation of the axioms must only precise their *formal semantics*, which constraint the interpretation of the conceptual primitives, without forcing their *operational semantics*, which fix the way the axioms are used in a KBS to reason [9]. For example, the axiom *"the enemy of my enemy is my friend"* can be used to produce knowledge (*i.e.* to deduce, when there exists an enemy of one of my enemies, that he is my friend) or to check assertions (*i.e.* to check that any enemy of one of my enemies is my enemy). An axiom can also be automatically applied by the system or explicitly applied by the user. The combination of these two criteria produces four different contexts of use for an axiom: (i) *inferential implicit* to automatically produce new knowledge from given facts, (ii) *inferential explicit* to allow the user to produce new assertions from given facts, (iii) *validation implicit* to automatically check a fact base, and (iv) *validation explicit* to allow the user to check a fact base at his request[11].

Using heavyweight ontologies in applications requires their *operationalization*, which consists in (1) specifying the way the axioms will be used to reason through a *scenario of use* and (2) transcribing the axioms in operational forms (rules and/or constraints) according to the adopted scenario of use. Because the operational semantics of each axiom have to be specified, building a scenario of use consists in choosing, for each axiom, its *context of use* which defines the way the axiom will be used[12]. For a given ontology, each scenario of use (*i.e.* a set of contexts of use) leads to a different operational ontology, as shown in figure 12.

Operationalizing a domain ontology corresponds to building a KBS which can be used to reason on facts on the domain. For example, operationalizing OntoFamily O_1 in an inferential scenario of use produces an operational ontology appropriated to deduction: given few persons, linked only by descendants links, the KBS can automatically deduce parents links, brotherhood links and other family links. Another operationalization, in a validation

[11] Note that "deduction vs validation" and "implicit vs explicit" contexts of use are fine-grained examples of knowledge uses. At a more general level of granularity, a scenario of use can specify the reasoning mechanism used in a KBS (deduction, abduction, induction), or the goal of the KBS (*e.g.* teaching system or corporate memory management).

[12] According to the structure of the axiom and to the context of use, the operational form can be a rule, a constraint, or a set of rules and constraints (*cf.* [9] for more detail about this transformation process).

scenario of use, can be used to check a fact base of family relationships, in order to evaluate its consistency, its completeness and its conciseness.

The OntoFamily O_1, represented in OCGL, includes 3 concepts, 31 binary relations, 11 axiom schemata and 18 axioms. Operationalizing O_1 for automatically completing a set of facts, *i.e.* in an inferential implicit scenario of use, leads to an operational ontology which includes the same terminological knowledge (3 concepts and 31 relations) but 39 rules and 7 constraints. The operational ontology automatically generated can be used, for example, to deduce, from a graph that represents facts which only deals with direct parent relations (father and mother), all the family relationships such as brotherhood relations, grandfather and grandmother relation, niece, nephew, uncle, aunt, etc. (*cf.* [**?**] for more detail).

6.2 Reasoning domain ontologies at the meta-level

6.2.1 Operationalization at the meta-level

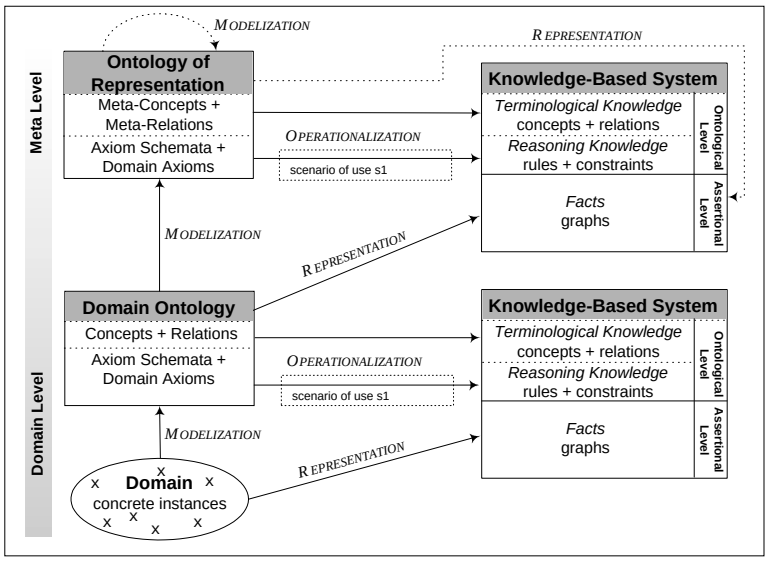

Figure 13: Overview of the interactions between Domain Ontology, Ontology of Representation and KBS.

Since domain ontologies are conceptual representations of a domain, their operationalization produces operational ontologies that enable reasoning on domain facts. In the same way, reasoning on ontologies themself can be done by operationalizing the representation ontology on which they are based, *i.e.* a meta-ontology. More precisely, if we consider ontologies expressed in the OCGL language (for example), operationalizing the ontology of the OCGL language produces operational ontologies that permit to reason about the first ontologies. The ontology of OCGL, called MetaOCGL, represents knowledge about the OCGL language, the primitives of the language, and their semantics expressed through axioms. Operationalizing MetaOCGL consists in choosing the way the axioms will be used to reason about an ontology expressed in OCGL, for example OntoFamily O_1. To complete O_1, by automatically adding subsomption links, or by propagating inherited properties, for example, MetaOCGL have to be operationalized in an inferential scenario of use. To validate OntoFamily according to the OCGL formal semantics, MetaOCGL have to be operationalized in a validation scenario of use.

Figure 13 shows the interactions that exist between Domain Ontology, Ontology of Representation and KBS. It also underlines the three main activities related to the integration of ontologies into KBS: *Modelization*, *Operationalization* and *Representation*.

At the domain level, an ontology (called Domain Ontology in figure 13) of a particular domain (called Domain in the figure) is built via a *modelization* process. Reasoning about facts on this domain in a KBS is allowed by operationalizing the ontology according to a particular scenario of use which describes the way the axiomatic part

of the ontology is used in the KBS. Then, the generated operational ontology can be used to reason about facts which are representations of instances of the domain. To sum-up, the *Modelization* of a domain leads to a domain ontology including *Concepts*, *Relations* and *Axioms* (both axiom schemata and domain axioms). The *Operationalization* of a domain ontology leads to the development of the ontological level of a KBS, including *Terminological Knowledge* (concepts and relations) and *Reasoning Knowledge*, *i.e.* rules and constraints corresponding to the operational forms of the axioms in the context of use which has been choosen. Finally, the *Representation* of a domain leads to the construction of the *Assertional Level* of the KBS, *i.e.* facts which are defined according to the *Terminological Knowledge*, and which are manipulated by the *Reasoning Knowledge*.

This three-step process (*Modelization*, *Operationalization*, *Representation*) can also be applied at the meta-level (*cf.* figure 13). The Ontology of Representation modelizes the language used to express the Domain Ontology. This ontology of representation is also expressed with the considered language. It can be operationalized in a KBS, and the generated operational ontology enables reasoning on the Domain Ontology. In this KBS defined at the meta-level, a fact is the representation of a particular domain ontology, for example a graph which represents OntoFamily O_1 in MetaOCGL. Because the ontology of representation is a meta-representation, modelizing this ontology in the same language produces the same ontology of representation. But this ontology can be represented as a fact in a KBS which implements an operational version of it, in order to reason on the ontology of representation itself.

6.2.2 Operationalization of MetaOCGL: an application to ontology evaluation

In order to use the MetaOCGL ontology for ontology evaluation (which includes verification, validation and assessment activities [11]), it is necessary to operationalize it in a validation and explicit scenario of use, *i.e.* all the axioms are used to validate a fact base. In the example of the figure 14, this fact base is the graph which represents an extract of the OntoFamily O_1. An error has been voluntarily introduced in the signature of the *"aunt(Woman,Universal)"* binary relation: this relation is a sub-relation of the *"relation_involving_a_Man(Woman,Human)"* relation. So, the signature of *"aunt"* is not in conformity with those of *"relation_involving_a_Man"*. The application of the signature conformity axiom (*cf.* figure 8), in a validation context of use, reveals the problem: the dark part of the graph is those which corresponds to the breaking of the axiom.

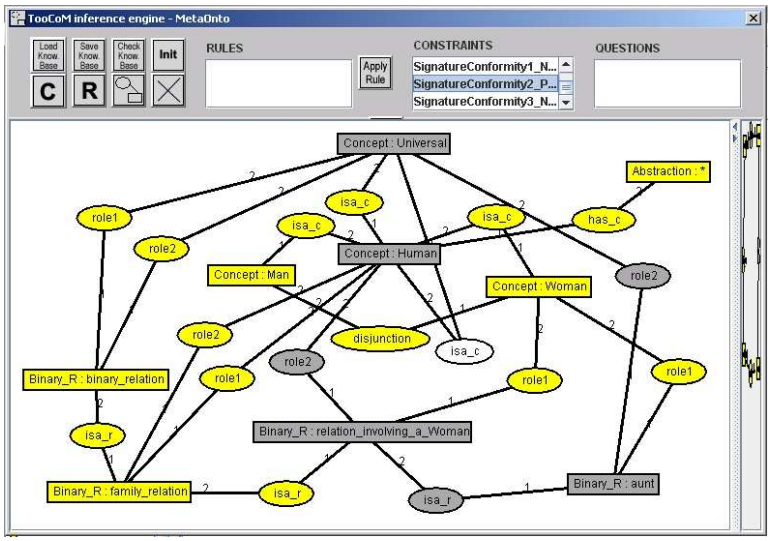

Figure 14: Operationalization of MetaOCGL for ontology verification.

Note that our approach allows the knowledge engineer to explicitly define, through the definition of axioms at the meta-level, the criteria used to evaluate the content of ontologies in terms of consistency, completeness and conciseness. This declarative definition of criteria at the conceptual level increases both the portability and the modularity of the evaluation criteria, which, in most of the similar works, are directly hard-coded in the tools.

6.3 Comparison between ontology engineering and model engineering

In this section, we have presented a new approach for reasoning domain ontologies at the meta-level, approach which mainly relies on the application of ontology operationalization mechanisms to an ontology of representation. We have also illustrated the relevance of our work in the context of two core questions for semantic interoperability: ontology evaluation and ontology matching.

In the field of model engineering, the OMG has defined a complete architecture called MDA (Model Driven Architecture) [19]. In the MDA, a specific model (level M1) captures each aspect of a system, a meta-model (level M2) captures a model and a meta-meta-model (level M3) captures a meta-model (*cf.* figure 15). For example, an UML model modelizes an application, UML modelizes this UML model and the MOF modelizes UML. The M0 level is the "real world", that is the applications in software engineering. A similar architecture can be considered in the field of ontology engineering: an ontology (level M1) is a model of a knowledge domain, a meta-ontology (level M2) is a model of ontology and a meta-meta-ontology (level M3) is a model of meta-ontology. For example, OntoFamily O_1 is a model of the family relationship domain, OCGL is a model of O_1, and Meta-OCGL is a model of OCGL.

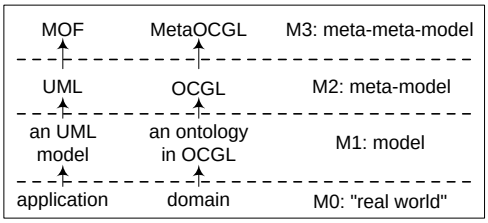

Figure 15: Standard OMG layered organization illustrated with UML and comparison with ontology engineering and OCGL.

What we claim is that reasoning the M1 level (for ontology validation or verification, ontology querying, ontology mapping, etc.) can be done by using terminological and axiomatic representations at the M3 level. This approach, based on a specific operationalization process (which can be compared to a transformation in model engineering), provides a more portable and modular method for reasoning domain ontology than methods which are only defined at the M2 level and thus totally specific to a predefined knowledge representation language. This approach can also increase the efficiency of ontology mapping, by enabling axiom comparison and primitive matching at the meta-level, independently from syntactical considerations.

Moreover, a link can be etablished between our approach in ontology engineering and model engineering techniques. Ontologies currently evolve from lightweight ontologies, used as simple descriptions of domains, to heavyweight ontologies, used to reason on domains. In a similar way, models evolve from descriptions of systems to sources for automatic application building [13]. In this context, ontology engineering and model engineering both focus on using more and more abstract representations, in order, on the one hand, to improve model or ontology building and (re)using and, on the other hand, to increase their independence from platforms.

7 Discussion

7.1 Related work

7.1.1 Related work in Ontological Engineering.

Currently, a lot of tools that deal with finding correspondences between ontologies are proposed [14].

The first way to classify these tools is to consider the objective which is pursued: (1) merging two ontologies to create a new one (*e.g.*, IPROMPT [16] or OntoMerge [4]), (2) defining a transformation function that transforms one ontology into another (*e.g.*, OntoMorph [1]) or (3) defining a mapping between concepts in two ontologies by finding pairs of related concepts (*e.g.*, ANCHORPROMPT [16], GLUE [3], S-MATCH [10], FCA-Merge [23] or ASCO [24]). Our work is dedicated to the latter objective by considering the mapping between relations in

addition. Note that although we are able to compare two axioms structurally, we have not yet considered the semantic mapping between axioms.

Another way to categorize the tools is to consider the type of input on which the tool relies in its analysis and which it requires: (1) class names and natural-language definitions, (2) class hierarchy and properties (*e.g.*, ANCHORPROMPT, GLUE), (3) instances of classes (*e.g.*, FCA-Merge) or (4) descriptions of classes (*e.g.*, S-MATCH). Some tools consider several inputs: for instance, ASCO [24] uses both (1), (2) and (4) [24]. Our approach is based on both (2) class hierarchy and properties, (4) descriptions of classes and, in addition, a new type of input: axioms.

7.1.2 Related work in Database Schema Integration.

As recalled in [16], database schemas are similar to ontologies: both are structural representations of knowledge. The difference often lies in scale rather than substance: database schemas in practice use much fewer modelling primitives than ontologies and are often smaller than ontologies. Just as with ontologies, researchers are often faced with the problem of finding correlation between different schemas. However, the more common approach in the database-schema research is to develop mediators rather than find point-to-point correlations (see [18] for a survey of approaches to Automatic Schema Matching).

The main difference between schema-matching and ontology-matching approaches remains the number of knowledge-modelling primitives on which the analysis relies. Schema-matching approaches look only for matches between concepts, whereas in ontology matching finding correspondences between relations (or properties) is just as important. In addition, in the context of our work, we consider axioms for finding correspondences between concepts and relations (and in future, between axioms too): this approach (which can be compared by using integrity constraints for finding correlations), is not yet developed in the field of database schema matching.

7.2 Similarity measures

In [5], a similarity stack is provided in order to classify the different measures that can be used to perform ontology matching. This stack is composed of five levels: the *Entities* level, the *Semantic Nets* level, the *Description Logics* level, the *Restrictions* level and the *Rules* level.

For the first three levels, the authors provide similarity measures which of course differ according to semantic complexity of the level which is considered. Thus, for the first level, the measure only considers the labels of the entities. The rule based on this measure is as follow: labels are human identifiers (names) for entities and are normally shared by a community of humans speaking a common language, so if labels are the same, the entities are probably also the same. The second level takes the properties (*i.e.* the relations between concepts) into account: if the properties of two concepts are equal, the concepts are also equal. The third level is based on the *ISA* relationship that can be used to define a hierarchy of concepts and/or a hierarchy of properties (*i.e.* to define taxonomies). An example of a rule based on this *ISA* relationship is: if super-concepts are the same, the actual concepts are similar to each other. This level also considers the instances: concepts that have the same instances are the same.

However, for the *Restrictions* level and *Rules* level, no measure is proposed. Explanations given by the authors are the following: "*the features like algebraic properties or equivalence/disjointness are not sufficiently used by the community to be considered as a material for similarity measure; for the Rules level, there has not been sufficient research and practical support for the Rule Layer of the Semantic Web Layer Cake*".

Our work must be considered as an extension of this classification of similarity measures in the sense that it provides measures based on the axioms of the domain which include both the *Restrictions* level and the *Rules* level. However, as we claim that it is not possible to consider rules and constraints at the ontological level (rules and constraints only exist at the operational level, *cf.* [9] for the justification of this point of view), we propose to modify the stack by merging the two levels *Restrictions* and *Rules* into only one: the *Axioms* level.

Then, to compare the different ways to compute the distance between two entities of two ontologies, [7] provides the following taxonomy:

- *Terminological (T)*: comparing the labels of entities. Two approaches are distinguished: (1) string-based considering only string structure dissimilarity (TS) and (2) lexicon-based (TL) including the relationships found in a lexicon (*i.e.* considering synonyms as equivalent and hyponyms as subsumed);

- *Internal structure comparison (I)*: comparing the internal structure of the entities (*e.g.* the value range or the cardinality);
- *External structure comparison (S)*: comparing the relations of the entities with other entities. Two approaches are distinguished: (1) taxonomical structure (ST) comparing the position of the entities within a taxonomy and (2) external structure comparison with cycles (SC), an external structure comparison robust to cycles;
- *Extensional comparison (E)*: comparing the known extension (*i.e.* the instances) of the entities;
- *Semantic comparison (M)*: comparing the interpretations (or more exactly the models) of the entities.

The originality of our work is that, by considering the axioms of the domain, we deal with the *Semantic Comparison (M)*. As recalled in [7], the only work which also considers (M) is the algorithm of [10] which uses a complete prover to decide subsumption or equivalence classes given initial equivalence of some classes and analysis of the relationships in the taxonomy. What differentiates our work from this approach is that we do not only consider the *ISA* relationship for interpretating the entities; we consider all the semantic structure of the entities including axiom schemata (*ISA* relationship, algebraic properties, etc.) and domain axioms which are the main material for fixing the semantic interpretation of the entities. For the same reason, our work is complementary from the ANCHORPROMPT system [16] which uses the *ISA* graph structure to find correlations between concepts, and do not consider the axioms expressed in PAL (Protégé Axiom Language). Note that like the other methods (*cf.* [7] for a detailed comparison), we also consider I, ST. We do not consider E.

7.3 Limitations

First, our approach suffers from the same limitation as the graph-based algorithm underlying the tool ANCHORPROMPT[16]: it does not work well when the two ontologies have major modelling differences in their internal structure. However, in such a context, our approach will be more efficient than ANCHORPROMPT, due to the axioms which are generally independent from the concept and relation modelization choices (topologically speaking).

Secondly, evaluation of primitive matching is simply based on the arithmetic sum of weights. The significance of each axiomatic representation for comparing primitive is then taken into account, but not the one of combination of representations. For example, a matching of two relations that are reflexive, symmetric and transitive is more significant than one of two relations that are reflexive, transitive and incompatible with another one, even if the sum $W_R + W_S + W_T$ is equal to $W_R + W_T + W_{Incomp}$, because the combination of reflexivity, symmetry and transitivity is the equivalence property, which is a very significant property. So, to improve the evaluation of primitive matchings, combinations of properties must be weigthed, and not only each property. The discrete Choquet integral [2] is exactly an aggregation operator which permits to weight the subsets of element of the sum, and not only each element. This operator is widely used in multicriteria decision systems and fuzzy systems [12], and it can be applied to our ontology matching problem in order to more precisely evaluate primitive matching.

Formally, a Choquet integral is defined with respect to a fuzzy measure. A fuzzy measure μ on a set C is a set function $\mu : \mathcal{P}(C) \rightarrow [0,1]$ satisfying $\mu(\emptyset) = 0$, $\mu(C) = 1$ and $\forall A, B \in \mathcal{P}(C), A \subset B \Rightarrow \mu(A) \leq \mu(B)$. The discrete Choquet integral of scores $x_1, ..., x_n$ with respect to the fuzzy measure μ is defined by $\sum_{i=1}^{n} x_{\sigma(i)}[\mu(A_{\sigma(i)}) - \mu(A_{\sigma(i+1)})]$ where σ is a permutation that order the elements $x_{\sigma(1)} \leq ... \leq x_{\sigma(n)}$ and $A_{\sigma(i)} = \{\sigma(i), ..., \sigma(n)\}$ ($A_{\sigma(n+1)} = 0$. We currently study the integration of this integral in our algorithm.

7.4 Semantic matching Versus Syntactic matching

As recalled in [10], syntactic matching differs substantially from semantic matching in that, instead of computing semantic relations between nodes, it computes syntactic similarity coefficients between labels, mainly in the [0,1] range. Possible semantic relations are: *equivalence*, *more general*, *less general*, *mismatch* or *overlapping*. Thus, for instance, the concepts of two nodes are equivalent if they have the same extension, they mismatch if their extensions are disjoint, and so on. In [10], they order these relations according to their binding strength, from the strongest to the weakest. Thus, equivalence is the strongest since the mappings tells us that the concept of the

second node has exactly the same extension as the first, more and less general give us a containment information with respect to the extension of the concept of the first node, mismatch provides a containment information with respect to the extension of the complement of the concept of the first node while, finally, overlapping does not provide any useful information, since we have containment with respect to the extension, of both the concept of the first node and its negation.

In the context of our approach, although we assign coefficients, we clearly concentrate on *semantic matching* in the sense that we calculate mappings by computing the semantic relations holding between the concepts (and not labels!) assigned to nodes. But we only focus on the *equivalence* semantic relation.

Then, according to J. SOWA [21], to integrate two ontologies means to derive a new ontology that facilitates interoperability between systems based on the original ontologies, and he distinguishes three levels of integration: *Alignment* - a mapping of concepts and relations to indicate equivalence, *Partial compatibility* - an alignement that supports equivalent inferences and computations on equivalent concepts and relations, and *unification* - a one-to-one alignement of all concepts and relations that allows any inference or computation expressed in one ontology to be mapped to an equivalent inference or computation in the other ontology. But what *equivalence* means? As recalled in [15], this is not a formally and consensually agreed term, neither do we have mechanisms for clarify this notion. Thus, in first-order logic, two concepts are equivalent if, and only if, they are first-order equivalent: this is the usual approach to formal semantic interoperability. In some works, as semantics is defined in terms of instances, two concepts are equivalent if, and only if, they share exactly the same instances (*e.g.* FCA-Merge).

In our work, two concepts are equivalent if, and only if, they are first-order equivalent.

8 Conclusion

In this paper, we have introduced a new ontology matching approach mainly based on the use of axioms. This approach has been defined in the context of a graph-based knowledge representation and graph-based reasoning mechanisms.

Our method has the advantage of incorporating most of the descriptive features of a heavyweight ontology into the matching process whereas most of the current methods cover only subsets of a lightweight ontology (mainly the hierarchy of concepts and their natural language expression). Of course, we are conscious that our method, although applicable, can not be very efficient in a context of lightweight ontologies. However, as demonstrated by the current challenge "Reasoning the Semantic Web", the need for developing heavyweight ontologies inevitably will increase in an immediate future. So, it seems interesting to focus on developping matching techniques dedicated to this type of ontology. This paper presents only preliminary work and some of the main issues we need to work on are: (1) to develop and to integrate in TOOCOM an efficient implementation of our approach, (2) to do a thorough testing of our ideas, (3) to compare our work to the other state of the art matching systems in our particular context dedicated to heavyweight ontologies (for this purpose, we plan to use the benchmark and the evaluation method proposed at EON'2004 - `http://km.aifb.uni-karlsruhe.de/ws/eon2004`) and to use the criteria proposed by the OntoWeb project [17] (pp 36-37)) and (4) to study how subsomption links and instances can be used in our matching method. We also study how to complement our approach with the reverse: from "using axioms to discover concept/relation matchings" to "using previously attested concept/relation matchings to discover axioms matchings".

References

[1] Hans Chalupsky. OntoMorph: A Translation System for Symbolic Knowledge. In *Principles of Knowledge Representation and Reasoning*, pages 471–482, 2000.

[2] G. Choquet. Theory of Capacities. *Annales de l'Institut Fourier*, 5:131–295, 1955.

[3] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Ontology Matching: A Machine Learning Approach. In *Handbook on Ontologies in Information Systems*, pages 397–416, 2004.

[4] D. Dou, D. McDermmot, and P. Qi. Ontology Translation on the Semantic Web. In *International Conference on Ontologies, Databases and Applications of SEmantics (ODBASE2003)*, pages 952–969, 2003.

[5] M. Ehrig and Y. Sure. Ontology Mapping - an integrated approach. In *Proceedings of the First European Semantic Web Symposium*, pages 76–91. Springer-Verlag (LNCS 3053), 2004.

[6] Esperonto-IST-2001-34373. State-of-the-art report: Indexing techniques, routing and peer-to-peer, similarity measures - deliverable 4.3. In *Esperonto Services. Application Service Provision of Semantic Annotation, Aggregation, Indexing and Routing of Textual, Multimedia, and Multilingual Web Content*, http://www.esperonto.net/, 2003.

[7] J. Euzenat and P. Valtchev. Similarity-based ontology alignment in OWL-Lite. In R. Lopez de Mantaras and L. Saitta, eds., *European Conference on Artificial Intelligence (ECAI'2004)*, pages 333–337. IOS Press, 2004.

[8] F. Fürst. TooCoM: a Tool to Operationalize an Ontology with the Conceptual Graph Model. In *Proceedings of the Workshop on Evaluation of Ontology-Based Tools (EON'2003) at ISWC'2003*, pages 57–70, 2003.

[9] F. Fürst, M. Leclère, and F. Trichet. Operationalizing domain ontologies: a method and a tool. In R. Lopez de Mantaras and L. Saitta, eds., *European Conference on Artificial Intelligence (ECAI'2004)*, pages 318–322. IOS Press, 2004.

[10] F. Giunchiglia, P. Shvaiko, and M. Yatskevich. S-Match: an Algorithm and an Implementation of Semantic Matching. In *Proceedings of the First European Semantic Web Symposium*, pages 61–65. Springer-Verlag (LNCS 3053), 2004.

[11] A. Gomez-Perez, M. Fernandez-Lopez, and O. Corcho. *Ontological Engineering*. Springer, Advanced Information and Knowledge Processing, 2003.

[12] M. Grabisch. Fuzzy integrals in multicriteria decision making. 69:279–298, 1995.

[13] J. Greenfield and K. Short. Software Factories: Assembling Applications with Patterns, Models, Frameworks and Tools. In *Proceedings of the 18th Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA'2003)*, pages 16–27, 2003.

[14] Y. Kalfoglou and M. Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18(1):1–31, 2003.

[15] Y. Kalfoglou and M Schorlemmer. Formal support for representing and automating semantic interorerability. In *Proceedings of the First European Semantic Web Symposium*, pages 45–60. Springer-Verlag (LNCS 3053), 2004.

[16] N. F. Noy and M. Musen. The PROMPT suite: Interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*, 59(6):983–1024, 2003.

[17] OntoWeb-IST-2000-29243. A survey on ontology tools - deliverable 1.3. In *Ontology-based information exchange for knowledge management and electronic commerce*, http://www.ontoweb.org/, 2002.

[18] E. Rahm and P.A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.

[19] R. Soley and the OMG staff. Model-Driven Architecture. 2000.

[20] J. Sowa. *Conceptual Structures : information processing in mind and machine*. Addison-Wesley, 1984.

[21] J. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing, 2000.

[22] S. Staab and A. Maedche. Axioms are objects too: Ontology engineering beyong the modeling of concepts and relations. Research report 399, Institute AIFB, Karlsruhe, 2000.

[23] Gerd Stumme and Alexander Maedche. FCA-MERGE: Bottom-up merging of ontologies. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI'2001)*, pages 225–234, 2001.

[24] Bach Thanh Le, Rose Dieng-Kuntz, and Fabien Gandon. On Ontology Matching Problems - for Building a Corporate Semantic Web in a Multi-Communities Organization. In *Proceedings of the 6th International Conference on Enterprise Information Systems (ICEIS'2004)*, volume 4, pages 236–243, 2004.

LABORATOIRE D'INFORMATIQUE DE NANTES-ATLANTIQUE

Axiom-based ontology matching: a method and an experiment

Frédéric Fürst, Francky Trichet

Abstract

Managing multiple ontologies is now a core question in most of the applications that require semantic interoperability. The Semantic Web is surely the most significant application of this report: the current challenge is not to design, develop and deploy domain ontologies but to define semantic correspondences among multiple ontologies covering overlapping domains. In this paper, we introduce a new approach of ontology matching named *axiom-based ontology matching*. As this approach is founded on the use of axioms, it is mainly dedicated to heavyweight ontology, but it can also be applied to lightweight ontology as a complementary approach to the current techniques based on the analysis of natural language expressions, instances and/or taxonomical structures of ontologies. This new matching paradigm is defined in the context of the Conceptual Graphs model (CG), where the projection (*i.e.* the main operator for reasoning with CG which corresponds to homomorphism of graphs) is used as a means to semantically match the concepts and the relations of two ontologies through the explicit representation of the axioms in terms of conceptual graphs. We also introduce an ontology of representation, called MetaOCGL, dedicated to the reasoning of domain ontology at the meta-level.

Categories and Subject Descriptors: I.2.4 [**Knowledge Representation Formalisms and Methods**]: Representation Languages

General Terms: Knowledge Engineering, Knowledge Representation, Ontology Matching

Additional Key Words and Phrases: Heavyweight Ontologies, Conceptual Graphs

LINA, Université de Nantes
2, rue de la Houssinière
B.P. 92208 — F-44322 NANTES CEDEX 3